Міністерство освіти і науки України
Національний технічний університет України “Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №6
З дисципліни «Технології розроблення програмного забезпечення»
Тема: **«ШАБЛОНИ «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator»»**
Powershell terminal

Виконав:
Студент групи ІА-22
Парій І. Р.

Перевірив:
Мягкий М. Ю.

Київ-2024

Зміст

Тема:
ШАБЛОНИ «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator»

Мета:
Ознайомитися з основними шаблонами проєктування, такими як «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator», дослідити їхні принципи роботи та навчитися використовувати для створення гнучкого та масштабованого програмного забезпечення.

Варіант
..8 Powershell terminal (strategy, command, abstract factory, bridge, interpreter, client-server)
Термінал для powershell повинен нагадувати типовий термінал з можливістю налаштування кольорів синтаксичних конструкцій, розміру вікна, фону вікна, а також виконання команд powershell і виконуваних файлів, а також працювати в декількох вікнах терміналу (у вкладках або одночасно шляхом розділення вікна).

Хід роботи

1. Реалізувати не менше 3-х класів відповідно до обраної теми

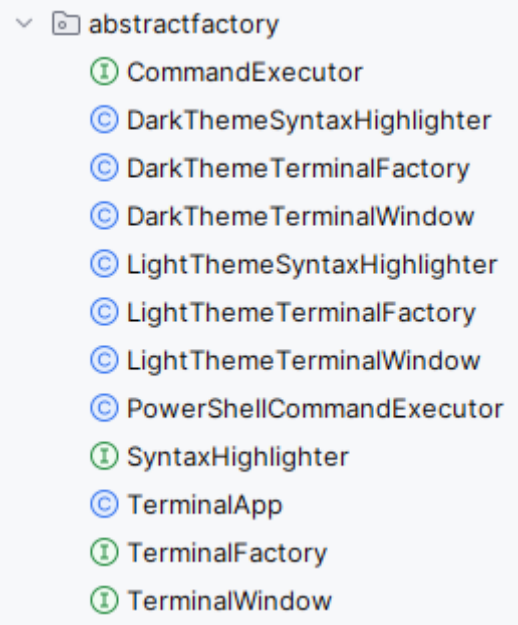

Рис. 1 — Структура проекту

У процесі виконання лабораторної роботи було створено 4 класи, які забезпечують реалізацію функціоналу PowerShell термінал (рис. 1). Наведемо детальний опис кожного з цих класів:

1. CommandExecutor
   - Відповідає за виконання команд PowerShell та запуск виконуваних файлів. Забезпечує взаємодію між терміналом і операційною системою для обробки введених користувачем команд.
2. DarkThemeSyntaxHighlighter
   - Реалізує підсвічування синтаксису для PowerShell у темній темі. Визначає кольори та стилі для ключових слів, змінних, рядків і коментарів.
3. DarkThemeTerminalFactory
   - Фабрика для створення терміналів із темною темою оформлення. Забезпечує конфігурацію за замовчуванням для фону, кольору тексту та інших елементів інтерфейсу.
4. DarkThemeTerminalWindow
   - Клас для створення вікна терміналу з темною темою. Підтримує налаштування розміру вікна, кольору фону та організації вікон/вкладок.
5. LightThemeSyntaxHighlighter
   - Реалізує підсвічування синтаксису для PowerShell у світлій темі. Аналогічний класу для темної теми, але з використанням більш світлих кольорів.
6. LightThemeTerminalFactory
   - Фабрика для створення терміналів зі світлою темою оформлення. Забезпечує необхідні параметри для кольорів і фону, щоб відповідати вимогам світлого дизайну.
7. LightThemeTerminalWindow
   - Клас для створення вікна терміналу зі світлою темою. Забезпечує такі самі функції, як і для темної теми, але з іншими налаштуваннями візуального оформлення.
8. PowerShellCommandExecutor
   - Спеціалізована версія CommandExecutor, орієнтована на виконання специфічних PowerShell команд. Додає підтримку функцій PowerShell, таких як обробка скриптів, модулів і змінних середовища.
9. SyntaxHighlighter

- Абстрактний базовий клас для підсвічування синтаксису. Визначає загальні методи та інтерфейси для темної та світлої версій підсвічування.

10. TerminalApp
    - Головний клас додатка, який забезпечує ініціалізацію та управління усіма компонентами терміналу. Координує роботу вікон, вкладок, налаштувань та виконання команд.
11. TerminalFactory
    - Абстрактна фабрика для створення об'єктів терміналів. Дозволяє створювати як темні, так і світлі термінали, базуючись на виборі користувача або налаштуваннях системи.
12. TerminalWindow
    - Абстрактний базовий клас для вікон терміналу. Визначає функціонал для управління розміром вікна, фоном, кількістю вкладок та інших елементів інтерфейсу.

Ці класи разом формують архітектуру терміналу, яка дозволяє створити багатофункціональний інструмент із гнучким налаштуванням інтерфейсу та можливістю одночасної роботи у декількох вікнах або вкладках.

2. Реалізувати один з розглянутих шаблонів за обраною темою

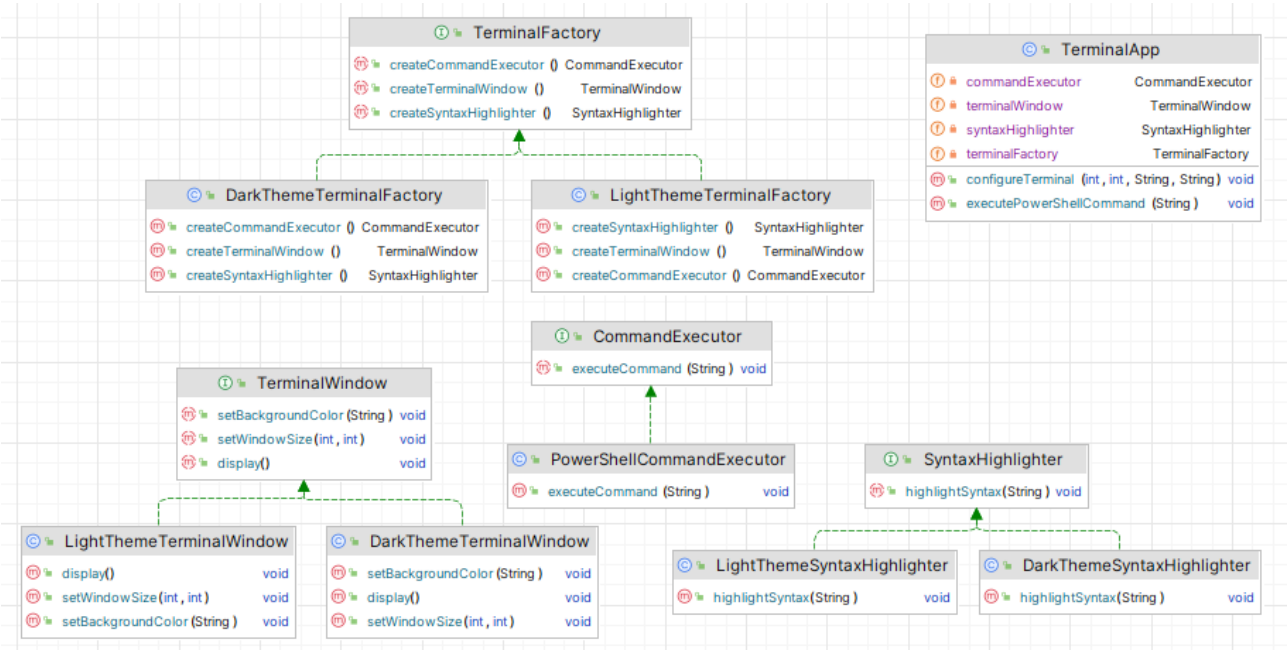

Рис. 2 — Діаграма класів

У проекті додатку PowerShell терміналу було реалізовано патерн Abstract Factory для створення компонентів інтерфейсу терміналу, таких як вікно терміналу, підсвітка синтаксису та виконання команд, адаптованих під різні теми оформлення (темну і світлу).

Як це реалізовано:

1. TerminalFactory — це абстрактна фабрика, яка визначає інтерфейси для створення основних компонентів терміналу: вікна терміналу, командного виконавця та підсвічувача синтаксису.
2. LightThemeTerminalFactory і DarkThemeTerminalFactory — це конкретні фабрики, що реалізують інтерфейс TerminalFactory для створення компонентів, специфічних для світлої або темної теми.
3. Основні компоненти терміналу включають:
   - TerminalWindow — вікно терміналу, що може відображатися з різними параметрами.
   - CommandExecutor — клас для виконання команд в PowerShell.
   - SyntaxHighlighter — підсвічувач синтаксису, що змінює вигляд залежно від теми (світла або темна).
4. TerminalApp — основний клас додатку, який взаємодіє з фабриками для створення необхідних компонентів залежно від теми. Він абстрагує логіку від користувача щодо того, яку тему використовувати, що спрощує підтримку та зміни.

Проблеми, які вирішує:

1. Легкість зміни інтерфейсу
   - Застосування патерну дозволяє легко змінювати зовнішній вигляд терміналу, не змінюючи основний код. Якщо користувач змінює тему, достатньо підключити іншу фабрику, і всі компоненти терміналу будуть автоматично налаштовані під нову тему.
2. Гнучкість та масштабованість
   - Додаток легко масштабувати, оскільки можна додати нові теми або інші налаштування терміналу, просто створивши нову фабрику та відповідні класи для кожного компонента. При цьому код, що використовує фабрику, залишиться незмінним.
3. Інкапсуляція створення об'єктів
   - Клієнтський код (в нашому випадку клас TerminalApp) не потребує знання про конкретні класи, які він використовує для створення

компонентів інтерфейсу. Вся логіка створення об'єктів інкапсульована в фабриках, що робить код чистішим і менш залежним від деталей реалізації.

Переваги використання патерна Abstract Factory:

1. Зручність у підтримці та розширенні
   - Якщо з'явиться потреба додати нові стилі або налаштування терміналу, це можна зробити без модифікації основного коду програми.
2. Зменшення кількості залежностей
   - Оскільки створення об'єктів зосереджене в конкретних фабриках, це зменшує кількість прямих залежностей в клієнтському коді.
3. Єдність стилю компонентів
   - Abstract Factory гарантує, що всі компоненти терміналу (вікно, підсвітка, виконання команд) будуть створені в єдиному стилі, забезпечуючи узгодженість теми оформлення.

Застосування

```
public class TerminalServiceImpl {
    5 usages
    private TerminalFactory terminalFactory;

    no usages
    public TerminalServiceImpl(TerminalFactory terminalFactory) { this.terminalFactory = terminalFactory; }

    no usages
    public void setTerminalFactory(TerminalFactory terminalFactory) { this.terminalFactory = terminalFactory; }

    no usages
    public void configureTerminal(int width, int height, String bgColor, String code) {
        TerminalWindow terminalWindow = terminalFactory.createTerminalWindow();
        SyntaxHighlighter syntaxHighlighter = terminalFactory.createSyntaxHighlighter();

        terminalWindow.setWindowSize(width, height);
        terminalWindow.setBackgroundColor(bgColor);
        terminalWindow.display();
        syntaxHighlighter.highlightSyntax(code);
    }

    no usages
    public void executePowerShellCommand(String command) {
        CommandExecutor commandExecutor = terminalFactory.createCommandExecutor();
        commandExecutor.executeCommand(command);
    }
}
```

Рис. 3 — Застосування патерну

У класі TerminalServiceImpl використовується патерн Abstract Factory для абстракції процесу створення різних компонентів терміналу (вікно терміналу, підсвічувач синтаксису, командний виконавець). В залежності від переданої фабрики (TerminalFactory), цей клас може створювати відповідні компоненти для конкретної теми (світлої або темної). Клас надає методи для конфігурації терміналу (встановлення розмірів вікна, кольору фону, підсвічування коду) та для виконання команд PowerShell, використовуючи відповідні об'єкти, що створюються фабрикою. Це дозволяє змінювати тему інтерфейсу або поведінку терміналу без зміни логіки самого класу.

Висновки:

Ми використали патерн Abstract Factory, щоб спростити створення різних компонентів терміналу для різних тем (світла і темна) та забезпечити гнучкість в їх виборі. Це дозволило ізолювати логіку створення об'єктів, пов'язаних із темами, в окремі фабрики, уникнути дублювання коду і зменшити кількість умовних операторів. У результаті ми досягли легкості зміни теми або додавання нових, забезпечивши масштабованість та спрощену підтримку проєкту без зміни основної бізнес-логіки.